Claude Code

Cẩm nang Claude Code cho startup

Năm quy tắc giúp startup ra sản phẩm như đội lớn gấp mười lần

Chia sẻ từ nhà sáng lập của:

- Artemis Security
- Cainex
- Clay
- ClickHouse
- Commure
- Crosby
- Emergent
- Harvey
- Heidi
- Higgsfield
- Omni
- Parahelp
- Translucent
- Zingage

Phiên bản tiếng Việt · Biên dịch & Việt hóa bởi Phong Hồ

Những công ty thuần AI ở tuyến đầu

Muốn nhìn trước tương lai của công việc thì cứ hỏi các startup xem hôm nay họ đang vận hành ra sao. Chúng tôi đã hỏi.

Chúng tôi trò chuyện với hơn một chục startup tăng trưởng nhanh về cách họ dùng công cụ lập trình bằng agent để làm sản phẩm và mở rộng công ty. Những startup này đang viết lại luật chơi: ai được quyền làm ra sản phẩm, cái gì bị bỏ đi, và làm sao tạo được bánh đà giữa cách bạn xây và thứ bạn xây.

Và họ ra sản phẩm với tốc độ của những tổ chức lớn gấp mười lần.

ClickHouse ra thêm 30% tính năng

Omni tăng năng suất kỹ thuật 2-3 lần

Clay tự động hóa 100% quy trình phân loại lỗi

Artemis Security đẩy hơn 6.000 PR mỗi tuần

Trong cẩm nang này, chúng tôi đi sâu vào cách triển khai riêng của từng tổ chức, để rút ra những quy tắc giúp họ ra sản phẩm nhanh và giữ được lợi thế cạnh tranh.

Qua đó, chúng ta cũng dần có câu trả lời cho câu hỏi: một tổ chức xây toàn bộ vòng đời phát triển sản phẩm bằng Claude Code ngay từ đầu thì sẽ trông như thế nào?

Mẹo: Bạn chỉ quan tâm những việc làm được ngay? Cuối cẩm nang có một danh sách kiểm gom lại toàn bộ mẹo kỹ thuật quan trọng của từng chương.

01

Ai cũng ra được sản phẩm

Lập trình bằng agent hạ thấp rào cản, nên người hiểu vấn đề nhất có thể tự làm ra bản sửa đầu tiên.

QUY TẮC 1

Ai cũng ra được sản phẩm

Lập trình bằng agent hạ thấp rào cản để nhân sự không rành kỹ thuật cũng làm được sản phẩm. Với Claude Code, bạn tạo ra tính năng chạy được mà không cần thạo một ngôn ngữ lập trình hay biết dùng IDE.

Mads Lunau Liechti
Parahelp

Mads Lunau Liechti, đồng sáng lập Parahelp, kể: "Không chỉ kỹ sư ra sản phẩm nhiều hơn hẳn, mà người không rành kỹ thuật (như tôi) cũng bỗng nhiên tự thay đổi được giao diện và cải tiến sản phẩm."

Với nhà sáng lập startup, lợi ích quá rõ. Thứ nhất, họ không có quân số như đối thủ lớn nên buộc phải "cả nhà cùng xắn tay". Nhưng thứ họ cần không chỉ là thêm sức người: những thành viên không rành kỹ thuật này còn mang theo hiểu biết chuyên môn về lĩnh vực.

Theo **Ryan Daniels, đồng sáng lập kiêm CEO của Crosby**, "Claude Code đã thay đổi ý nghĩa của việc làm luật sư ở Crosby. Chính luật sư mới có góc nhìn sản phẩm sắc nhất, vì họ là người dùng. Nhìn họ trổ tài thật đã mắt."

Ryan Daniels
Crosby

Dr. Thomas Kelly
Heidi

Chúng tôi nghe điều tương tự từ **Dr. Thomas Kelly, đồng sáng lập kiêm CEO của Heidi**: "Với chúng tôi, Claude Code giải được bài toán tam sao thất bản. Trước đây một ý tưởng mới đi qua đội ngũ thế này: người có ý tưởng kể cho PM, PM kể cho designer, designer kể cho kỹ sư... và thế nào cái hồn của ý tưởng cũng rơi rụng đâu đó trên đường.

Đến lúc ra được thì thứ làm ra thường chẳng còn giống điều người kia hình dung. Mà mất hàng tuần. Claude Code rút gọn chuỗi đó lại. Người thực sự hiểu vấn đề có thể tự mở một PR, rồi kéo designer và kỹ sư vào đúng những chỗ cần chuyên môn của họ."

Nói "ai cũng ra được sản phẩm" thì nghe rất hợp để đăng LinkedIn, nhưng thực tế chạy ra sao? Đội marketing đi duyệt pull request à? Đội pháp lý ngồi mò từng commit tìm con test chập chờn à?

Câu trả lời chúng tôi nhận được: vẫn có phân công lao động. Người làm marketing vẫn lo marketing, lập trình viên vẫn lo lập trình. Nhưng bước đầu tiên và quan trọng nhất, đưa một ý tưởng thành bản mẫu chạy được, đi từ 0 đến 1, thì mở cho tất cả mọi người.

Chúng tôi cũng thấy những startup hiệu quả nhất đều tạo ra cơ chế để các đóng góp này thành hệ thống, thay vì phó mặc cho may rủi hay nhiệt huyết cá nhân.

Tạo kết nối

Đặt kỳ vọng nhân viên phải dùng AI là một chuyện. Cho họ quyền dùng Claude Code cùng những công cụ họ cần lại là chuyện khác.

"Chúng tôi không hề né chuyện [để nhân sự không rành kỹ thuật đóng góp], mà còn đi thẳng vào đó. Quan điểm của chúng tôi là mọi vị trí đều đang trở thành vị trí kỹ thuật, vì bạn có thể tự viết phần mềm cho công việc của mình... nên chúng tôi tuyển những người thích mày mò, thích tự tay dựng nên thứ gì đó," **Kareem Amin, đồng sáng lập kiêm CEO của Clay**, cho biết.

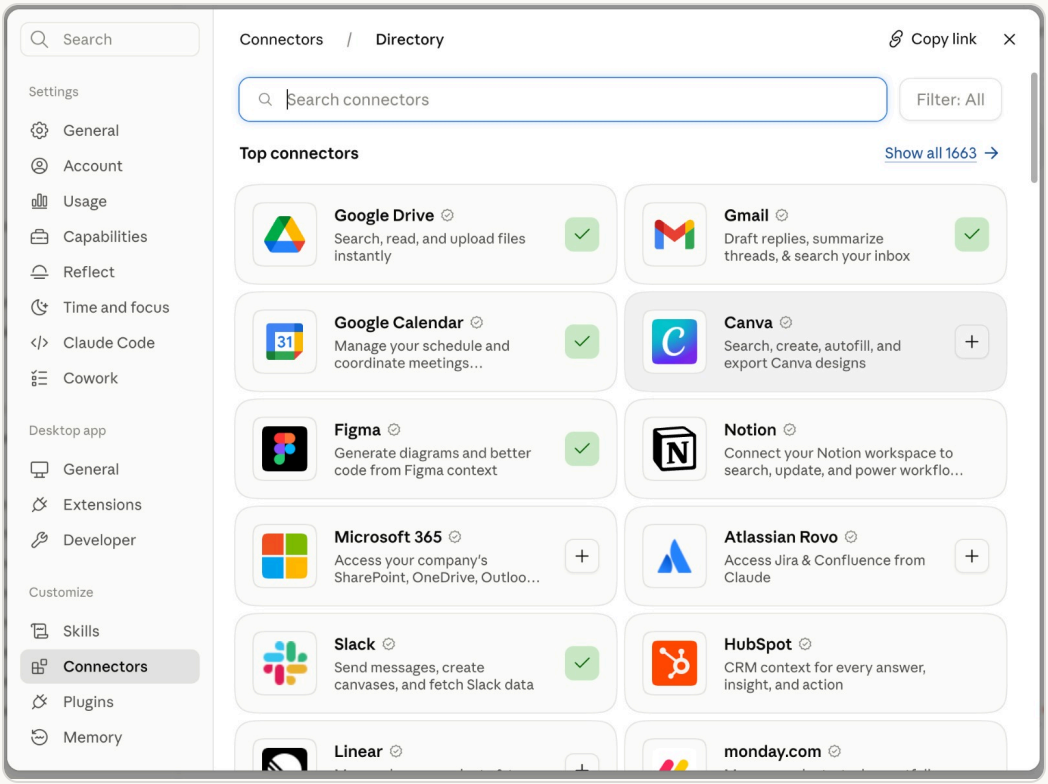
Kareem Amin
Clay

Ở Crosby, đội ngũ không kéo luật sư đến với Claude Code. Họ mang Claude Code đến với luật sư, bằng cách nối nó vào đúng những công cụ và hệ thống mà luật sư đã quen dùng mỗi ngày.

Mẹo: Claude không hiểu được thứ nó không nhìn thấy. Một trong những cách hiệu quả nhất để tăng giá trị của Claude là nối nó vào các nguồn dữ liệu gốc và những công cụ đội bạn dùng hằng ngày.

MCP là chuẩn mã nguồn mở để kết nối AI với công cụ, cho Claude Code truy cập vào công cụ, cơ sở dữ liệu và API của bạn. Hễ thấy đội mình đang copy rồi dán thông tin từ một công cụ nào đó vào Claude, hãy nghĩ đến việc thêm kết nối này.

Nối qua CLI có thể tiết kiệm token hơn khi đã sẵn một công cụ dòng lệnh trưởng thành (`gh`, `kubectl`, `bq`, `psql`) và bạn muốn Claude làm việc trên đúng nguồn dữ liệu mà kỹ sư của bạn đang dùng.



Thư mục kết nối MCP trong ứng dụng Claude Code trên máy tính.

Trình diễn trong buổi standup

Đến một lúc nào đó, ý tưởng cần có cơ hội được xếp thứ tự ưu tiên, để nguồn lực của tổ chức đưa nó ra thị trường. Với product manager thì con đường đó rõ ràng, vì đấy là việc của họ. Với nhân sự không rành kỹ thuật thì không rõ như vậy.

Clay tổ chức buổi duyệt hằng quý, nơi các bản mẫu được xem xét và có thể vào lộ trình sản phẩm chính thức. Nhờ vậy mà một thành viên đội đưa sản phẩm ra thị trường ở Clay đã dựng được một agent tự chạy: nó vào website của bạn, điền biểu mẫu thu khách hàng tiềm năng, bấm giờ xem bao lâu thì có phản hồi, chấm điểm trải nghiệm rồi xuất ra báo cáo.

Omni có hẳn một kênh Slack riêng cho các bản mẫu do Claude tạo ra, với đóng góp từ tất cả mọi người, kể cả nhân sự kỹ thuật cấp cao. Họ còn thực hành hệ quả của "ai cũng ra được sản phẩm", đó là "ai cũng nói chuyện với khách hàng".

Chris Merrick
Omni

Chris Merrick, đồng sáng lập kiêm CTO của Omni, cho biết: kỹ sư vốn không mấy hào hứng với các cuộc gọi khách hàng, nhưng Omni vẫn cố ý đưa họ ra trước khách hàng, vì như vậy vòng phản hồi khép lại nhanh hơn.

Chia sẻ skill

Ranh giới giữa "ai cũng ra được sản phẩm" và "chắp vá" rất mỏng. Bản mẫu tính năng, dù đến từ ai, vẫn phải ghép được vào một sản phẩm liền mạch. Đây là lúc skill phát huy tác dụng: những tệp hướng dẫn dùng lại được, ghi lại chuẩn mực và bối cảnh của đội bạn, giúp việc phát triển không lệch nhau ngay cả khi quy trình ngày càng mở cho nhiều người.

"Bất kỳ ai trong đội cũng có thể dùng Claude Code để phác thảo thành phần sản phẩm, tài liệu marketing hay nội dung slide, lấy design system của chúng tôi làm chuẩn. Phần AI chạm tới sản phẩm thì phải vượt một ngưỡng cao hơn nhiều, và Claude Code giúp chúng tôi đạt ngưỡng đó chính xác hơn," **Dr. Thomas Kelly, Heidi**, nói.

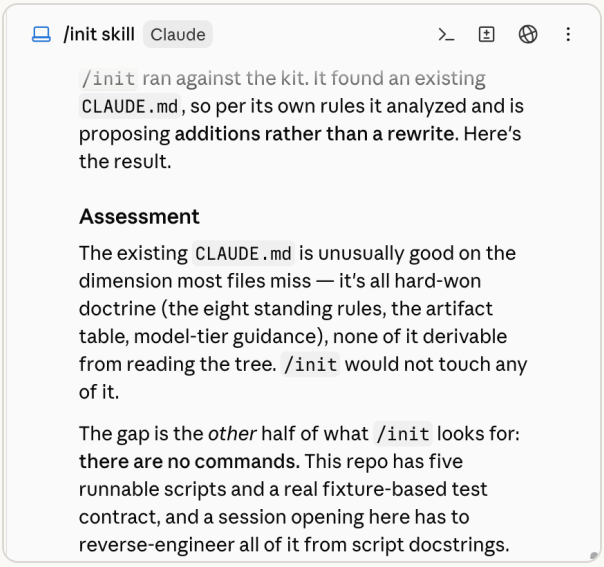
Mukund Jha
Emergent

Skill cũng giúp lập trình viên mới và nhân sự không rành kỹ thuật nhập cuộc và bắt nhịp nhanh. **Mukund Jha, đồng sáng lập kiêm CEO của Emergent**, mô tả cách làm này: "...chúng tôi còn có một repo GitHub chứa các skill của Claude Code, đóng vai trò kho kiến thức chung để khởi động nhanh một phiên Claude Code với những thông tin sẵn có của Emergent như vị trí cơ sở dữ liệu [và kho dữ liệu], một phần lược đồ [schema], bối cảnh chung của công ty.... Thay vì cố cho thật hoàn hảo, chấp nhận sống chung với vài tệp bối cảnh hơi cũ cũng không sao, miễn là agent kiểm lại nhanh được và tự chỉnh cho đúng."

Jack O'Hara
Translucent

"Kỹ sư của chúng tôi dùng Claude Code để dựng một chợ nội bộ gồm các agent chuyên biệt, sắp theo vai trò, nên kỹ thuật, triển khai và kinh doanh mỗi bên đều có công cụ làm đúng theo cách họ thực sự làm việc," **Jack O'Hara, nhà sáng lập kiêm CEO của Translucent**, cho biết.

Mẹo: Skill có thể chia sẻ toàn công ty qua một thư mục chung, để cách làm hay của một người lập tức truyền sang người khác. Dùng tệp CLAUDE.md trong từng thư mục con của repo cho các quy ước viết mã riêng của thư mục đó, áp dụng mọi lúc. Dùng skill cho những quy trình thao tác gọi ra khi cần. Đọc thêm: Điều hướng Claude Code: khi nào dùng CLAUDE.md, skill, hook và subagent.



`/init` là một skill phổ biến, tạo ra tệp CLAUDE.md để Claude nắm được các lệnh build, quy ước và cấu trúc dự án của bạn.

02

Tự động hóa phần việc buồn tẻ

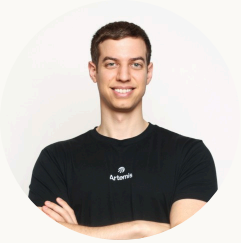
Agent lo 80% phần việc máy móc trong cả vòng đời, để kỹ sư dành thời gian cho những ca thật sự cần phán đoán.

QUY TẮC 2

Tự động hóa phần việc buồn tẻ

Từ buổi đầu cách mạng công nghiệp, công ty nào cũng tìm cách dùng công nghệ để làm việc hiệu quả hơn. Nhưng những startup này tách khỏi phần còn lại nhờ tốc độ và độ sâu khi áp dụng.

Các nhà sáng lập này coi AI là thành phần cốt lõi trong sứ mệnh của họ. Nhiều người nói thẳng: agent lo 80% phần việc máy móc, để kỹ sư dành thời gian cho những ca thật sự cần phán đoán.



Shachar Hirshberg
Artemis Security

"Ai cũng đang đua làm sản phẩm AI. Rất ít người chịu xây lại cách công ty mình vận hành. Vế thứ hai mới là thứ mở khóa lớn hơn. Artemis Security vận hành như một công ty thuần AI, không phải một công ty tình cờ có dùng AI. Điều đó đẩy tốc độ của chúng tôi lên hẳn và giúp khách hàng chặn tấn công ở tốc độ máy," **Shachar Hirshberg, đồng sáng lập kiêm CEO của Artemis Security**, nói.

Cụ thể, chúng tôi thấy AI được cài sâu vào các giai đoạn trong vòng đời phát triển phần mềm (SDLC) hơn hẳn nơi khác, và nhiều agent được dựng riêng để ôm trọn một việc lặp đi lặp lại từ đầu đến cuối. Cùng xem vài ví dụ cho cả hai.

Vòng đời phát triển phần mềm thuần AI

Nhiều startup trong bài đã có cách riêng để đội ngũ bắt nhịp nhanh với quy trình lập trình bằng agent.

Ví dụ ở Emergent, Mukund kể: "ngay ngày đầu, người mới dựng xong toàn bộ môi trường lập trình chỉ bằng cách chỉ cho Claude đúng tệp markdown cần đọc. Trong lúc đó, hễ Claude gặp chỗ nào hỏng hay lỗi thời, nó cập nhật luôn tệp ấy."

Mẹo: Code Review (bản xem trước phục vụ nghiên cứu) là dịch vụ nhiều agent do Claude Code quản lý. Nó tự chạy một lượt rà soát trên các PR trong những repo bạn bật. Bạn có thể tự sửa theo phát hiện đó rồi push, hoặc khép vòng bằng cách bình luận @Claude ngay tại phát hiện (nếu bạn đã cài và cấu hình GitHub Actions).

Dấu	Mức độ	Ý nghĩa
🔴	Quan trọng	Lỗi nên sửa trước khi gộp nhánh
🟡	Vặt	Vấn đề nhỏ, nên sửa nhưng không chặn
🟣	Có sẵn	Lỗi đã có trong mã nguồn, không phải do PR này sinh ra

Code Review gắn nhãn mức độ cho từng phát hiện.

Kỹ sư phải bắt nhịp nhanh, vì các đội này ra sản phẩm rất nhanh.

Tanay Tandon
Commure

Theo **Tanay Tandon, CEO kiêm nhà sáng lập Commure**: "Kỹ sư ở đây điều phối cả đội agent, sửa xong lỗi dữ liệu trên môi trường thật ngay trong ngày phát hiện, và chạy nhiều PR song song cùng lúc. Một kỹ sư đã chạy một đợt việc khoảng 13 ticket với các subagent Claude chạy song song, mỗi subagent ôm một ticket và PR của nó."

Ở những tổ chức này, Claude Code không chỉ giúp sinh mã mà còn rà soát mã. "Chúng tôi chạy rà soát mã tự động dựa trên bộ khung kỹ thuật và tuân thủ đã được kiểm duyệt, đánh dấu các vấn đề nghiêm trọng và chuyển đề xuất sửa tới đúng người rà soát, trước khi bất cứ thứ gì được đưa ra," **Dr. Kelly của Heidi** cho biết.

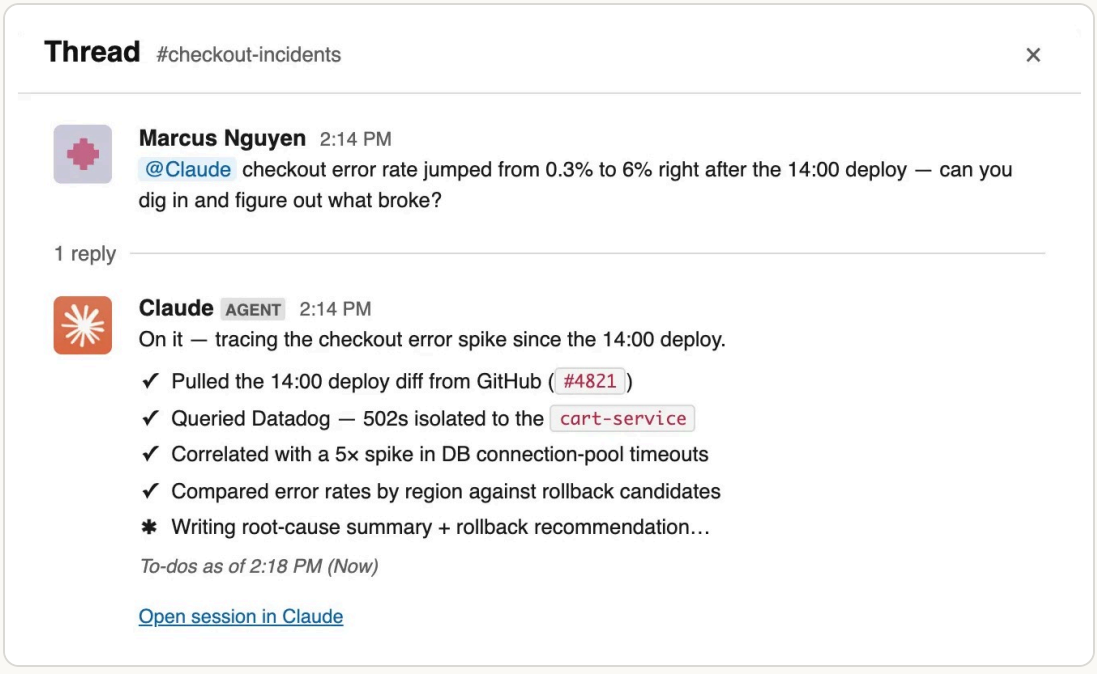
Một số tổ chức còn tự dựng agent riêng cho việc rà soát mã, kiểm thử và CI. Các startup này đặt rất nhiều tâm sức vào việc dựng loop, tức các vòng lặp agent tự chạy, chứ không chỉ triển khai mã.

"[Agent] tôi thích nhất là "Translucent code reviewer": nó tỏa ra khắp một thay đổi, soi từ nhiều góc, rồi tổng hợp kết quả đúng kiểu một kỹ sư cấp cao của chúng tôi vẫn làm, nhưng nhanh hơn bất kỳ ai một mình làm nổi," **Jack, nhà sáng lập Translucent**, nói.

Clay "...đã dựng một agent lo... việc phân loại lỗi, từ lượt rà đầu tiên cho tới đề xuất thay đổi mã để sửa," **Kareem** cho biết.

Mẹo: Suốt mấy tháng qua, Claude Tag là người trực đầu tiên xử lý các lỗi CI/CD ở Anthropic. Trong mọi sự cố gần đây có báo cáo, Claude đều là người viết báo cáo tình hình đầu tiên, thường công bố phân tích đầu tiên trong vòng 15 phút.

Claude Tag có tài khoản dịch vụ riêng và quyền dùng những công cụ mà một kỹ sư CI của Anthropic cần, như Datadog hay Grafana. Các chỉ dẫn thường trực nằm trong tệp markdown dưới dạng skill, được commit vào một repo GitHub. Nhờ vậy nhiều người trong đội cùng chỉnh sửa dần được, và chúng tôi quản lý thay đổi y như quản lý mã.



Claude Tag nhận một luồng trực sự cố trên Slack và báo tiến độ ngay trong kênh.

Alexey Milovidov
ClickHouse

Rõ nhất là ở ClickHouse. **Alexey Milovidov, đồng sáng lập kiêm CTO**, cho biết công ty cơ sở dữ liệu này đã biến gần như mọi giai đoạn trong vòng đời phát triển phần mềm thành một loop tự chạy. Hai agent chuyên dụng, một chuyên sửa test chập chờn và một chuyên tìm chỗ thiếu kiểm thử, giờ là người đóng góp nhiều thứ 2 và thứ 3 cho repo ClickHouse. Một nhóm agent khác lo phần vận hành, và chính đội ngũ dùng Claude Code để dựng và cải tiến những agent đó.

Tăng tốc quy trình bằng agent

Một khuôn mẫu nữa lặp lại đều đặn: các startup này không chỉ dùng loop agent trong Claude Code để tăng tốc phát triển, mà còn tạo ra agent để tăng tốc những quy trình lặp đi lặp lại và thường rất buồn tẻ.

Đó thường là việc thường nhật, giao đi để dồn tâm trí cho lợi thế cạnh tranh, quan hệ khách hàng và tăng trưởng doanh thu. Một trong những quy trình được Claude tăng tốc phổ biến nhất là phân tích dữ liệu tự phục vụ.

Gần như công ty nào cũng có sẵn một quy trình để ra quyết định nhanh dựa trên dữ liệu mới, kể cả dữ liệu phi cấu trúc. Đó chính là thứ nuôi những cú chuyển hướng, vốn sống còn với một startup.

Ví dụ, Clay dựng một agent phân tích nội bộ, còn Heidi dùng Claude Code để phân loại phản hồi của khách hàng và bác sĩ cùng với dữ liệu sử dụng, nhằm lọc ra những tín hiệu đáng chú ý cho sản phẩm.

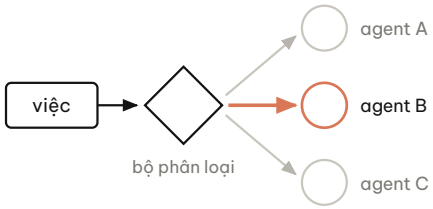
Cả ClickHouse và Omni đều bán ra sản phẩm có sẵn kiểu phân tích dữ liệu bằng AI này bên trong, tất cả đều chạy bằng Claude.

Vài ví dụ khác: tóm tắt hàng nghìn văn bản pháp lý bằng subagent (Crosby), quét dữ liệu yêu cầu bảo hiểm để đánh dấu bất thường trên nhiều cơ sở (Commure), và liên tục đào dữ liệu tài chính bệnh viện để tìm dấu hiệu cảnh báo mà không đội phân tích nào kịp phát hiện (Translucent).

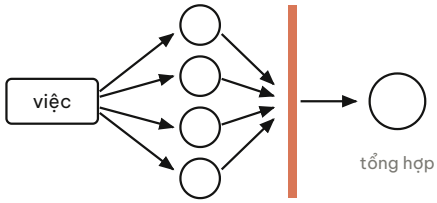
Mẹo: Dynamic workflows, tức quy trình động, cho phép tỏa ra nhiều subagent để phân tích khối lượng dữ liệu lớn song song, hoặc để rà soát đối kháng công việc của một agent khác. Khi dùng model như Claude Opus hay Claude Fable, hãy nói "tỏa ra nhiều subagent" hoặc "dùng một workflow".

Sáu khuôn mẫu quy trình

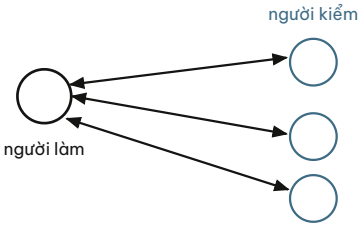
1 Phân loại rồi hành động



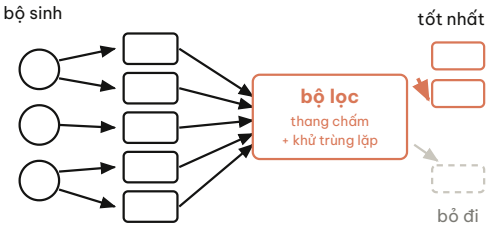
2 Tỏa ra rồi tổng hợp



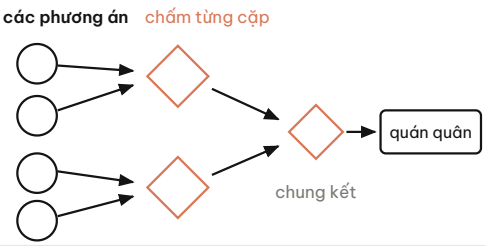
3 Kiểm chứng đối kháng



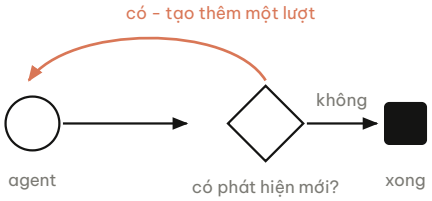
4 Sinh ra rồi lọc



5 Đấu loại



6 Lặp đến khi xong



03

Tin, nhưng phải kiểm chứng

Bạn không thể tự động hóa một quy trình nếu chưa có cách đáng tin để theo dõi và kiểm chứng kết quả.

QUY TẮC 3

Tin, nhưng phải kiểm chứng

Quy tắc này là hệ quả tất yếu của Quy tắc 2: Tự động hóa phần việc buồn tẻ. Bạn không thể tự động hóa một quy trình nếu chưa có cách đáng tin để theo dõi và kiểm chứng kết quả.

Dan Shiebler
Artemis Security

Dan Shiebler, đồng sáng lập Artemis Security, nói tốc độ triển khai tăng lên như vậy chỉ có được... "vì chúng tôi đã đầu tư rất sâu vào hạ tầng kiểm thử, cách tổ chức mã nguồn và hệ thống tri thức của đội, để agent làm trọn được từ đầu đến cuối. Đây chính là bánh đà chúng tôi dựng cùng Claude: sắp xếp mã nguồn, kho tri thức và đội ngũ cho đúng, thì mọi đóng góp đều cộng dồn lên nhau."

Victor Hunt
Zingage

Victor Hunt, đồng sáng lập kiêm CEO của Zingage, kể: "Giai đoạn đầu chúng tôi cho Claude toàn quyền, và nó làm đúng cái AI vẫn làm: ra mã nghe rất hợp lý, rất nhanh. Vấn đề là nó trôi khỏi kiến trúc của chúng tôi theo những cách nhìn thì đúng mà thực ra không đúng. Thế là chúng tôi... viết ra từng điều bất biến. Cách chúng tôi đặt vấn đề. Cái gì buộc phải đúng trong mọi hoàn cảnh. Cách chứng minh một thứ chạy được, thay vì tin vào một câu trả lời nghe đầy tự tin. 567 dòng ghi lại cách đội này suy nghĩ."

Mẹo: Đặt những thứ không được phép đổi vào tệp CLAUDE.md ở thư mục gốc của repo. Claude đọc tệp này ở đầu mỗi phiên, nên các quy tắc kiến trúc, ranh giới bảo mật và những điều không thương lượng sẽ đi theo mọi phiên làm việc.

Nói cho rõ: không startup nào trong số này để agent gộp thẳng vào nhánh main rồi cầu trời cho êm. Nhiều công ty hoạt động trong ngành bị quản lý rất chặt và cần khung quản trị mạnh. Cainex là ví dụ đặc biệt rõ về việc kết hợp agent với các bước kiểm tất định, tức chạy lần nào cũng cho kết quả như nhau, để đọc hồ sơ bệnh án và sinh ra mã điều hướng việc thanh toán viện phí.

Uriah Israel
Cainex

"Trong mã hóa y tế, một mã sai không phải lỗi gõ nhầm. Đó là một sự kiện về thanh toán và tuân thủ. Riêng sự thật đó chi phối toàn bộ cách chúng tôi xây," **Uriah Israel, đồng sáng lập kiêm CTO của Cainex**, nói.

"Đây là loop mà Claude Code chạy cho chúng tôi. Một agent xử lý một lô hồ sơ, rồi kiểm toán viên của chúng tôi soát kết quả trên một ứng dụng nội bộ. Họ không chỉ nhìn thấy mã. Họ nhìn thấy cả lập luận của model, và nhận xét lên cả hai.... Mọi thứ đều được ghi phiên bản và truy vết được," anh nói.

"Rồi Claude Code tiếp quản. Nó đọc thẳng từ cơ sở dữ liệu: các dự đoán ban đầu, cùng mọi chỉnh sửa và nhận xét. Mỗi chỉnh sửa được gắn nhãn theo loại mã liên quan, nên Claude Code biết mình đang xem lỗi chẩn đoán, lỗi thủ thuật hay một nhóm khác, và đi thẳng tới đúng phần hướng dẫn chi phối loại mã đó.

Từ đó, nó tìm ra đúng đoạn hướng dẫn đã sinh ra lỗi rồi sửa lại, hoặc viết hướng dẫn mới nếu ca đó thật sự chưa từng gặp. Mọi thay đổi đều thực hiện trên một bộ hướng dẫn có ghi phiên bản, và được kiểm lại trên chính những hồ sơ đã sai. Quy tắc chúng tôi bắt buộc tuân theo: sửa nguyên tắc, đừng sửa ví dụ," anh nói tiếp.

"Rồi tới bước kiểm ngược. Một hồ sơ có thể có nhiều cách mã hóa đều chấp nhận được, nên không thể so khớp chuỗi ký tự. Bước kiểm này kết hợp so khớp theo ngữ nghĩa với các bộ đáp án đã duyệt, cùng một trọng tài đặt câu hỏi: 'Đây là lỗi thật hay chỉ là một hướng đi khác vẫn đúng?', và Claude Code bổ sung thêm các phép so sánh của riêng nó.

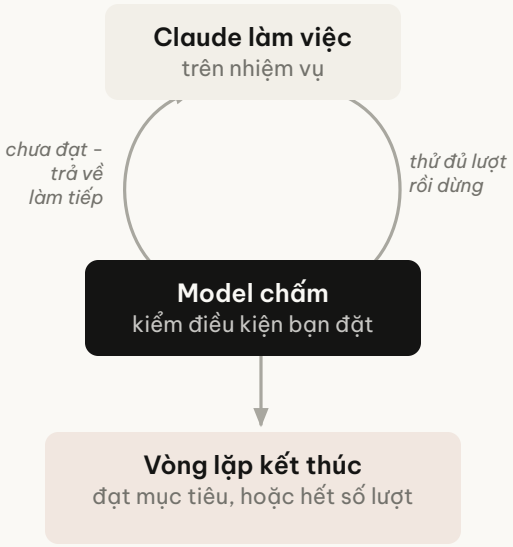
Nó đem thay đổi đang cân nhắc chạy trên một bộ chuẩn cộng thêm mẫu ngẫu nhiên, và nêu ra mọi chỗ bị thụt lùi trước khi thứ gì được đưa ra. Cái trả về là một danh sách ngắn: các chỗ đề xuất sửa, những hồ sơ nó không xử lý được, và những câu hỏi nó muốn được trả lời. Kỹ sư dành thời gian cho những ca thật sự khó, thay vì 80% phần việc máy móc," anh nói.

Có nhiều bài học chung mà nhà sáng lập rút ra được từ quy trình vốn rất đặc thù của ngành thanh toán y tế này.

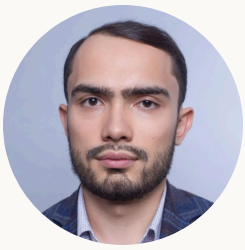
Ví dụ, Cainex dùng chuyên gia trong ngành để soát và uốn nắn lập luận của Claude một cách đều đặn, rồi đưa phần uốn nắn đó vào một loop tự cải thiện. Nhưng các chuyên gia này không ngồi sửa từng ví dụ một; hướng dẫn của họ được dùng như một phần của loop tự cải thiện. Như Uriah nói: "sửa nguyên tắc, đừng sửa ví dụ".

Mẹo: Loop là những agent lặp lại chu trình làm việc cho tới khi thỏa điều kiện dừng. Đây là cách hiệu quả để dùng Claude Code cho các việc cần tự chủ cao hoặc kéo dài. Bạn có thể dùng skill để định nghĩa tiêu chí agent phải đạt (càng rõ càng tốt) rồi để agent lặp cho tới khi tới đích.

Ví dụ, nhiều tổ chức tạo agent (hay loop) chuyên xử lý test chập chờn, vì điều kiện dừng vừa rõ vừa khép kín: agent tự kiểm được bản sửa của mình bằng cách chạy lại test cho tới khi đạt.



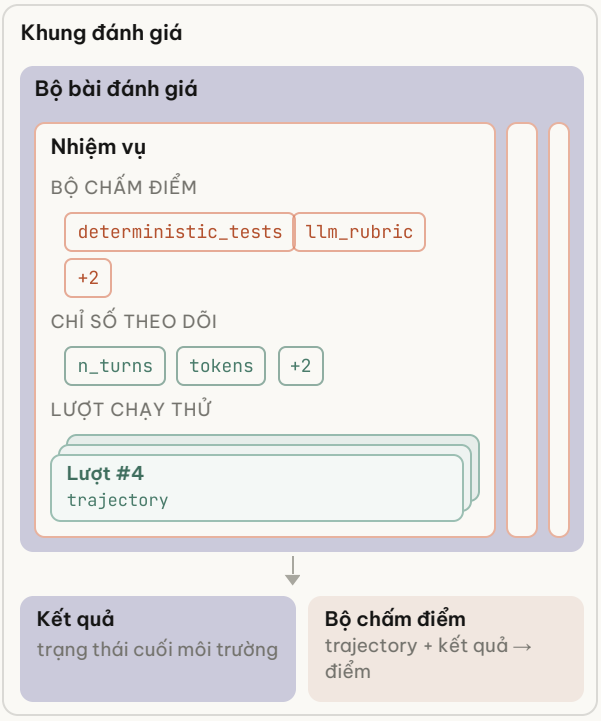
Bài học còn lại là sự tỉ mỉ khi duy trì một "bộ chuẩn" đánh giá thật mạnh, tức tập các cặp câu hỏi và câu trả lời đã được xác minh mà đội dùng để kiểm độ chính xác của agent. Mọi startup nên duy trì nhiều bộ eval, tức các bài kiểm tra đánh giá, cho những tình huống sử dụng chính, và cập nhật đều đặn để chặn tình trạng trôi lệch và đánh giá được các model ra sau.



Alex Mashrabov
Higgsfield

Alex Mashrabov, đồng sáng lập kiêm CEO của Higgsfield, nói: "[Claude Code] còn thay đổi hẳn cách chúng tôi theo kịp tốc độ ra model. Model video và ảnh mới xuất hiện liên tục. Mỗi cái đều cần skill mới, bài đánh giá mới, logic định tuyến mới và kiểm thử trên môi trường thật trước khi triển khai. Claude Code nén chu trình đó từ vài ngày xuống vài giờ, cho phép chúng tôi phát hiện vấn đề trên môi trường thật và triển khai bản sửa ngay trong cùng một phiên.... Khi phải cạnh tranh với những công ty đông gấp 10 lần, kiểu đòn bẩy đó thay đổi tất cả."

Mẹo: Khi mới bắt tay dựng agent, các đội đi được xa hơn nhiều người tưởng chỉ nhờ kiểm thử tay, tự dùng sản phẩm của mình và trực giác. Điểm gãy thường đến khi người dùng phản ánh rằng agent tệ đi sau khi sửa, còn cả đội thì "bay mù", không có cách nào kiểm ngoài đoán rồi thử. Đội không phân biệt được đâu là thụt lùi thật, đâu là nhiễu; không tự kiểm được thay đổi trên hàng trăm kịch bản trước khi đưa ra; cũng không đo được mức cải thiện. Đọc thêm: Giải mã eval cho AI agent.



Các thành phần của một bài đánh giá dành cho agent.

Ý cuối cùng Uriah nêu ra là quy trình này tốn công. "Lúc đầu không sạch sẽ như bây giờ đâu. Bản đầu tiên của chúng tôi bị quá khớp. Nó "sửa" bằng cách mã hóa cứng đúng ca đó, và chúng tôi cứ chất chồng miếng vá thay vì thông minh lên. Chúng tôi đổi cách làm: buộc phải rút ra nguyên tắc chung, và giới hạn số chi tiết cụ thể được phép lọt vào một thay đổi."

Mẹo: AI agent không tất định, nhưng rất nhiều công việc trong ngành bị quản lý chặt lại đòi hỏi quy trình phải chạy y hệt nhau mỗi lần. Claude Code có những tính năng giúp kết hợp trí thông minh ở tuyến đầu với quy trình tất định.

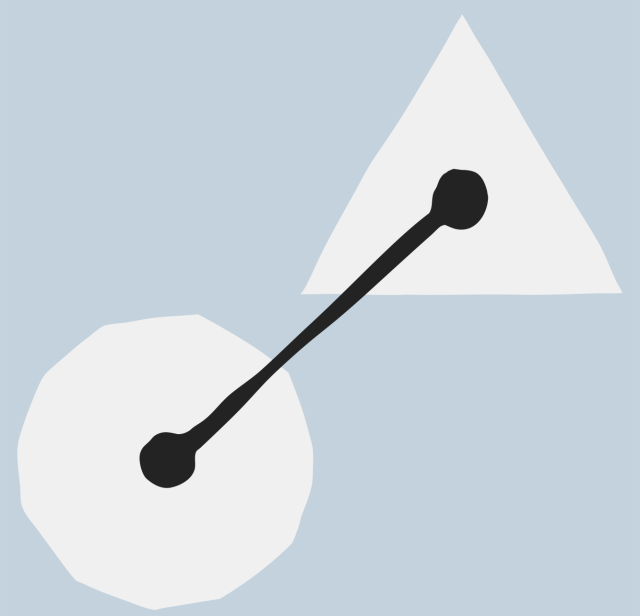
Hook là những lệnh do bạn tự định nghĩa, kích hoạt tại các mốc cố định trong vòng đời Claude Code, đóng vai trò cổng chặn cứng. Chúng chạy mọi lúc, bất kể model quyết định thế nào. Ví dụ: chặn thao tác ghi không qua được lint, bắt test phải đạt trước khi commit, hoặc lọc sạch khóa bí mật trước khi thứ gì rời khỏi sandbox.

Dynamic workflows điều phối subagent với trình tự tất định, cửa sổ ngữ cảnh riêng và mục tiêu tập trung. Lệnh `/goal` hữu ích cho việc dài và phức tạp, nơi Claude dễ tuyên bố xong việc quá sớm, thiên vị phát hiện của chính mình khi rà soát, và trôi khỏi mục tiêu ban đầu.

04

Xây để sẵn sàng xây lại

Năng lực của model liên tục thay đổi dưới chân các đội này, nên gần như không có gì được coi là vĩnh viễn.



QUY TẮC 4

Xây để sẵn sàng xây lại

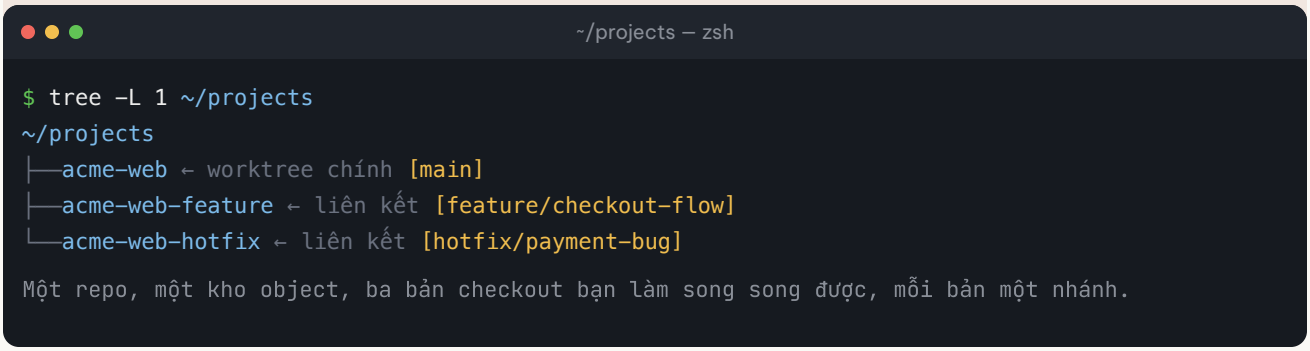
Nhiều startup thuần AI trong bài luôn ở trạng thái tự làm mới liên tục.

AI thường nằm ở lõi của cả thứ họ xây lẫn cách họ xây. Vì năng lực model tiến hóa không ngừng, những tính năng từng đột phá và cả phần khung đỡ quan trọng đều bị bỏ đi ngay khi chúng thành chi phí chìm. Nhiều tổ chức coi việc xây lại liên tục này là một phần lợi thế cạnh tranh của mình.

"Ở Clay, chúng tôi làm thế này: xây, rồi xây lại, rồi xây lại lần nữa. Đến lần thứ tư thì bạn đã biết hết những gì cần và làm đúng ngay. Nên không hẳn là chúng tôi vứt bỏ thứ gì. Chúng tôi chỉ xây lại thôi, và lần này thì rõ ràng hơn," **Kareem** nói.

"Một lần xây lại không kết thúc khi đường mới ra mắt. Nó kết thúc khi đường cũ biến mất. Trước đây việc dọn bỏ đường cũ luôn thua trong cuộc tranh giành thứ tự ưu tiên: vừa buồn tẻ vừa chẳng ra được tính năng nào," **Tanay, đồng sáng lập Commure**, nói. "Giờ một kỹ sư của Commure chỉ cần gọi một skill của Claude đại loại 'với mọi feature flag đã mở cho toàn bộ người dùng, hãy mở một PR gỡ nó và phần mã liên quan', rồi kỹ sư soát lại thứ trả về. Những đợt chuyển đổi từng ngốn rất nhiều công sức lập trình giờ chỉ là một bản kế hoạch và một lượt tỏa việc, xong trong vài giờ."

Mẹo: Dùng git worktree để chạy một lần xây lại trên bản sao tách biệt của repo, trong khi bản hiện tại vẫn nguyên vẹn. Claude Code có thể tạo giúp bạn: bạn có v2 chạy song song với v1, chạy eval trên cả hai, và chỉ gộp khi bản mới thắng. Đây chính là thứ khiến "xây bốn lần" trở nên rẻ.



Mỗi worktree liên kết là một thư mục thường với nhánh riêng đã checkout; cả ba dùng chung một kho object .git trong acme-web.

Kareem cũng mô tả một phần lợi thế phòng thủ của Clay chính là khả năng liên tục xây lại, tiến hóa và tạo ra những loop tự cải thiện.

"Tôi nghĩ lợi thế phòng thủ của bất kỳ công ty nào lúc này là nó phải tự cải thiện được. Clay là một cỗ máy doanh thu tự học. Bạn càng dùng, chúng tôi càng biết ai là khách hàng tốt nhất của bạn, bạn nên nói gì, cái gì đã hiệu quả, cái gì không, và điều đó thay đổi theo thời gian," anh nói. "Cuộc đua thật ra là ai chạm tới kênh phân phối nhanh nhất... để bạn giúp được từng [khách hàng], và nhờ đó tự cải thiện."

Niko Grupen
Harvey

Tại một sự kiện Code with Claude hồi tháng 5/2026, **Niko Grupen, Trưởng bộ phận AI ứng dụng của Harvey**, nói về việc mỗi làn sóng năng lực mới của model (lập luận trỗi dậy, tự động hóa bằng agent, lập kế hoạch và điều phối) đều buộc nền tảng phải tái kiến trúc toàn bộ.

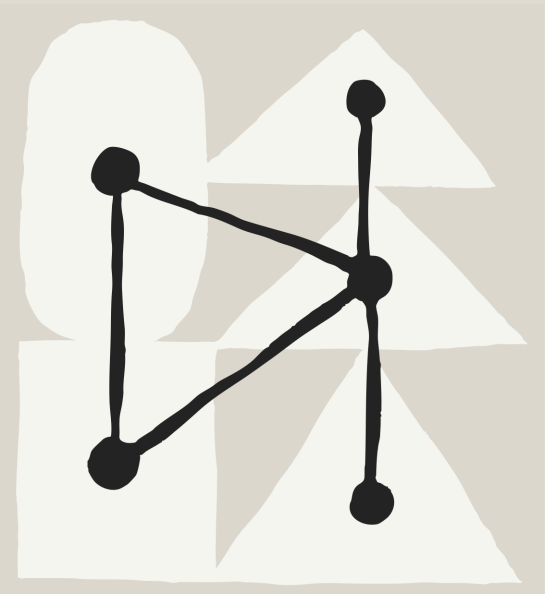
"Nếu sáu tháng trước bạn hỏi kiến trúc của chúng tôi trông thế nào, tôi sẽ trả lời khác hẳn hôm nay," anh nói. "Nếu chúng tôi không sẵn sàng nói 'Này, phải bỏ cái này đi và chuyển sang thuần agent' thì đơn giản là nền tảng của chúng tôi bây giờ không thể có những năng lực này."

Mẹo: Với những lần viết lại không đơn giản, hãy khởi động Claude Code ở plan mode, tức chế độ lập kế hoạch (dùng `--plan` hoặc bấm Shift+Tab). Claude sẽ khảo sát mã nguồn và đề xuất hướng xây lại trước khi viết bất kỳ dòng mã nào, để bạn duyệt hoặc chỉnh hướng. Đây là chỗ rẻ nhất để bắt được một lần xây lại sắp trôi khỏi kiến trúc của bạn.

05

Làm mẫu, tự dùng, đưa ra thật

Xây bằng AI giúp các startup này tạo ra sản phẩm AI đột phá. Đó là bánh đà nằm ở tâm quy trình của họ.



QUY TẮC 5

Làm mẫu, tự dùng, đưa ra thật

Nhiều startup trong bài có một bánh đà then chốt nằm ở tâm quy trình phát triển. Xây bằng AI giúp họ tạo ra những sản phẩm AI đột phá.

Khi lập trình viên nâng trình lập trình bằng agent, họ nắm chắc hơn năng lực của model và hiểu rõ hơn cách thiết kế harness, tức bộ khung vận hành quanh model, đang tiến hóa ở tuyến đầu. Từ đó họ mang cảm hứng ấy vào chính agent và sản phẩm của mình.

"Chúng tôi lấy cảm hứng từ cách [Anthropic] chọn dùng tệp thay vì embedding, và điều đó khiến chúng tôi mạnh dạn giữ mọi thứ đơn giản trong sản phẩm của mình. Chúng tôi tránh được cả đống phức tạp mà một pipeline RAG sẽ kéo theo," **Chris của Omni** nói. "Chúng tôi cũng thấy harness của Claude Code cho người dùng làm nhiều việc song song, nên đã đưa vài ý niệm đó vào giao diện của mình."

Cách này cũng giúp họ bám sát hiệu năng sản phẩm của chính mình.

"Vì công cụ dựng ứng dụng của chúng tôi cũng chạy model Anthropic bên dưới, nên hễ thấy một hành vi lạ trên sản phẩm... chúng tôi gỡ lỗi nhanh ngay trên máy bằng Claude Code để biết đó là do hành vi của model hay do harness. Việc này cải thiện chu trình phân loại sự cố của chúng tôi rất nhiều," **Mukund của Emergent** nói.

Khuôn mẫu chúng tôi nghe đi nghe lại là: dựng một agent nội bộ bằng Claude Code, cho nội bộ dùng thật (dogfood), rồi tùy phản hồi mà nâng nó thành sản phẩm cho khách hàng, thường là qua Claude API, SDK hoặc Claude Managed Agents.

"Chúng tôi tự dựng các agent AI [trong sản phẩm của mình] để các đội tương tác trực tiếp, gồm một agent trong bảng điều khiển SQL và một AI SRE. Chính chúng tôi dùng Claude Code để dựng và cải tiến những agent đó. Bộ công cụ tạo nên trải nghiệm AI cho khách hàng của chúng tôi, một phần, được xây bằng AI," **Alexey của ClickHouse** nói.

Danh sách kiểm

Cẩm nang này đi qua khá nhiều thứ. Dưới đây là những mẹo quan trọng gom lại trong một trang:

TÓM TẮT

Danh sách kiểm

Chương 1: Ai cũng ra được sản phẩm

- [ ] Claude không hiểu được thứ nó không nhìn thấy. Hãy nối nó vào các nguồn dữ liệu gốc và những công cụ đội bạn dùng hằng ngày, qua MCP hoặc CLI.
- [ ] Lập một chợ plugin của công ty để cách làm hay của một người lập tức truyền sang người khác qua một skill. Dùng tệp `CLAUDE.md` trong từng thư mục con của repo cho quy ước viết mã riêng của thư mục đó, áp dụng mọi lúc. Dùng skill cho quy trình thao tác gọi ra khi cần.

Chương 2: Tự động hóa phần việc buồn tẻ

- [ ] Bật Code Review (bản xem trước phục vụ nghiên cứu) trên một repo để tự rà soát PR.
- [ ] Đưa Claude Tag (bản beta công khai) vào quy trình trực sự cố CI/CD và phân loại lỗi.
- [ ] Dynamic workflows cho phép tỏa ra nhiều subagent để phân tích khối lượng dữ liệu lớn song song, hoặc rà soát đối kháng công việc của một agent khác.

Chương 3: Tin, nhưng phải kiểm chứng

- [ ] Đặt những thứ không được phép đổi vào tệp `CLAUDE.md` ở thư mục gốc của repo.
- [ ] Dùng loop, tức những agent lặp lại chu trình làm việc cho tới khi thỏa điều kiện dừng, cho các việc cần tự chủ cao hoặc kéo dài.
- [ ] Lập một quy trình để tạo và duy trì các bài đánh giá cho agent.
- [ ] Hook là những lệnh do bạn tự định nghĩa, kích hoạt tại các mốc cố định trong vòng đời của Claude Code và có thể làm cổng chặn cứng. Dùng hook khi một phần công việc bắt buộc phải tất định.

Chương 4: Xây để sẵn sàng xây lại

- [ ] Dùng git worktree để chạy một lần xây lại trên bản sao tách biệt của repo, trong khi bản hiện tại vẫn nguyên vẹn. Đây chính là thứ khiến "xây bốn lần" trở nên rẻ.
- [ ] Với những lần viết lại không đơn giản, hãy khởi động Claude Code ở plan mode (`/plan` hoặc bấm Shift+Tab). Claude sẽ khảo sát mã nguồn và đề xuất hướng xây lại trước khi viết dòng mã nào, để bạn duyệt hoặc chỉnh hướng. Đây là chỗ rẻ nhất để bắt một lần xây lại sắp trôi khỏi kiến trúc.

Startup ở tuyến đầu thì xây ở tuyến đầu

Những chia sẻ trên đến từ chính những người đang xây ở tuyến đầu cùng bạn, và chúng tôi mong bạn thấy chúng thiết thực, làm được ngay. Cộng đồng startup dùng Claude là nguồn cảm hứng, cách làm hay và lời khuyên không bao giờ cạn. Bạn có thể tham gia cộng đồng này bằng cách:

- Đăng ký nhận bản tin Startup và tham gia chương trình dành cho startup.
- Lưu lại lịch các buổi webinar Claude Code sắp tới.
- Tham dự một sự kiện gần bạn
- Theo dõi chúng tôi trên X và LinkedIn để nhận cập nhật mới nhất.
- Đóng góp trên Reddit và Discord.

Bản tiếng Việt được biên dịch và việt hóa bởi **Phong Hồ**.

✍️ Theo dõi thêm bài viết & tài nguyên hữu ích về AI → phongminhho.substack.com